INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

High order cross derivative computation for the differential cross section of double ionization of helium by electron impact

Isabelle Charpentier — Claude Dal Cappello

N° 5546

Avril 2005

Thème NUM

ISSN 0249-6399 ISRN INRIA/RR--5546--FR+ENG

High order cross derivative computation for the differential cross section of double ionization of helium by electron impact

Isabelle Charpentier*, Claude Dal Cappello †

Thème NUM — Systèmes numériques
Projet Idopt

Rapport de recherche n° 5546 — Avril 2005 — 23 pages

Abstract: The double ionization of an atom or a molecule is strongly dependent of the quality of the description of the initial state of the target. Recently, absolute measurements have been reported for the double ionization of helium by 5.6 keV electron-impact. Since the incident (and scattered) electron is very fast, one may apply the usual first Born approximation. Calculations with the first Born approximation lead to an overall magnitude that is about 50% larger than experiment when a simple (one term) Hylleraas wavefunction describes the initial state. Two numerical approaches are available to tackle an accurate (18 terms) Hylleraas wavefunction: a 6-dimensional numerical quadrature (expensive in computer time), or a 2-dimensional quadrature applied to high order cross derivatives (up to the order 9). Automatic differentiation tecnhiques allow for high order derivative computations. Nevertheless existing differentiation tools do not deal with codes written in complex arithmetics and explicit cross derivative computations.

This paper first describes the high order differentiation (based on recursive rules) and the extraction of cross derivatives. An operator oveloading library is constructed for the differentiation work. Numerical results, obtained at a lower cost than the sextuple integral, show the pertinence of our approach.

Key-words: cross derivatives, automatic differentiation, operator overloading, double ionization

* Projet Idopt, BP 53, F-38041 Grenoble cedex 9
† LPMC, 1, Bd. Arago, F-57078 Metz Cedex 3

Application de la differentiation automatique pour le calcul de la section efficace differentielle de la double ionisation de l'hélium par impact électronique

Résumé : La double ionisation d'un atome ou d'une molécule est fortement dépendante de la qualité de la description de l'état initial de la cible. Récemment, des mesures absolues ont été présentées pour la double ionisation de l'helium par des électrons de 5.6 keV. Comme l'électron incident est très rapide, l'approximation de Born au premier ordre peut être appliquée. Des calculs sous l'approximation de Born au premier ordre conduisent à une section efficace supérieure de 50% à celle donnée par l'expérience lorsqu'une fonction d'onde simple de type Hylleraas (avec un terme) décrit l'état initial. Deux approches numériques sont disponibles pour prendre en compte une fonction d'onde de Hylleraas précise (18 termes) : une intégration sextuple (coûteuse en temps de calcul), ou une double intégration appliquée à une expression contenant des dérivées d'ordre élevé (jusqu'à l'ordre 9). Les techniques de différentiation automatique permettent la dérivation de haut degré. Néanmoins les outils de différentiation ne traitent ni les codes écrits en arithmétique complexe, ni le calcul explicite des dérivées croisées.

Ce rapport décrit la différentiation de haut degré (basée sur des relations de récurrence) et l'extraction de dérivées croisées. Une bibliothèque surchargant les opérateurs est construite pour le travail de différentiation. Des résultats numériques, obtenus avec un coût moindre que celui de l'intégration sextuple, montre la pertinence de notre approche.

Mots-clés : dérivées croisées, différentiation automatique, surcharge d'opérateurs, double ionisation

1 Setting of the problems

Double ionisation of atoms or molecules by electron impact is of considerable interest in many fields of Physics such as plasma physics and astrophysics. Recently, [Lahmam-Bennani et al.(1999)] have performed measurements of fully differential cross section on helium. Their delicate experiment consists of detecting in coincidence the three electrons of the final state (one scattered and two ejected) by using three detectors. In these so-called $(e, 3e)$ experiments the directions and energies of all three electrons are completely determined, yielding the most detailed information possible about the double ionisation process (see the recent review paper [Berakdar et al.(2003)]). For a particular impact energy, the measurements give a fivefold differential cross section (FDCS) which depends upon the solid angles Ω_s, Ω_1 and Ω_2 for the scattered and the two ejected electrons, and on the energies E_1 and E_2 of ejected electrons. The incident (and scattered) electron is very fast and we use plane waves to describe it in the standard first Born formulation. Assuming that the mechanism of the double ionisation is limited to the shake-off (a unique interaction between the target and the incoming electron), this FDCS is computed as (atomic units are used throughout):

$$\frac{d^5\sigma}{d\Omega_s d\Omega_1 d\Omega_2 dE_1 dE_2} = \frac{k_1 k_2 k_s}{k_i}|M|^2, \tag{1}$$

where $\vec{k_i}$, $\vec{k_s}$, $\vec{k_1}$ and $\vec{k_2}$ denote, respectively, the momenta of incident, scattered, first ejected and second ejected electrons. The matrix element M is 9-dimensional integral such as:

$$M = \frac{1}{2\pi}\int \psi_f^*(\vec{r_1},\vec{r_2}) e^{i\vec{k_s}.\vec{r_0}} V \psi_i(\vec{r_1},\vec{r_2}) e^{i\vec{k_i}.\vec{r_0}} d\vec{r_0} d\vec{r_1} d\vec{r_2}, \tag{2}$$

where $V = -2r_0^{-1} + |\vec{r_0} - \vec{r_1}|^{-1} + |\vec{r_0} - \vec{r_2}|$ is the Coulomb interaction between the projectile and the helium atom, r_0 is the distance between the incident electron and the nucleus, r_1 and r_2 are the distances between one of the helium electron and its nucleus. The wavefunctions ψ_i and ψ_f are the solutions of the Schrödinger equation for the helium atom. No exact formulas exist for ψ_i and ψ_f.

The well-known Bethe transformation, $e^{i\vec{k}.\vec{a}}k^{-2} = 4\pi^{-1}\int e^{i\vec{k}.\vec{r}}|\vec{r} - \vec{a}|^{-1} d\vec{r}$, allows for the integration on $\vec{r_0}$. Thus the computation of (2) needs a 6-dimensional integral only. The method of [Brauner et al.(1989)] proves that the latter may be reduced to a double integral when the bound state wavefunction ψ_i is approximated by means of a Hylleraas-type wavefunction:

$$\psi_i(\vec{r_1},\vec{r_2}) = e^{-ks/2} \sum_{l,m,n;\ n\ even}^{0,\infty} c_{l,m,n} s^{l-m} u^{m-n} t^n, \tag{3}$$

where $s = r_1 + r_2$, $u = r_{12} = |\vec{r_1} - \vec{r_2}|$ and $t = -r_1 + r_2$. The initial state of an helium atom can be described with an accuracy of 5.10^{-5}% by a wavefunction [Kinoshita(1957)] with 39 parameters (it is not the case for its double continuum). The best approximation for the final state ψ_f is those of [Brauner et al.(1989)] which satisfies exact asymptotic boundary

collisions. One has:

$$\psi_f(\vec{r_1},\vec{r_2}) = \prod_{j=1}^{2} \frac{e^{i\vec{k_j}.\vec{r_j}}}{(2\pi)^{3/2}}\Gamma(1+\frac{iz}{k_j})e^{\pi z/2k_j}\ {}_1F_1(\frac{-iz}{k_j},1,-i(k_j r_j+\vec{k_j}.\vec{r_j})) \\ .\Gamma(1-i\alpha_3)e^{-\pi\alpha_3}\ {}_1F_1(i\alpha_3,1,-i(k_3 r_{12}+\vec{k_3}.\vec{r_{12}})) \quad (4)$$

where $\alpha_3 = |\vec{k_1}-\vec{k_2}|^{-1}$. Wave functions involving 6 parameters were used in [Brauner et al.(1989), Joulakian et al.(1992)], but as noticed in [Kheifets et al.(2002)] they are not accurate enough to conclude on the convergence of the calculations. Very recently, Jones and Madison [Jones & Madison(2003)] have performed such a calculation with a 20-parameter Hylleraas wavefunction with an expensive sextuple numerical integration.
The first aim of this paper is to show the feasibility of Brauner's method for more accurate wavefunctions as initial state. This will numerically prove its convergence, results being obtained at a lower cost than the sextuple integral.
Nevertheless, the gain in number of integrals has to be paid. In brief, Brauner's method is based on the computation of a term D corresponding to the simple wavefunction $\psi_i(\vec{r_1},\vec{r_2}) = e^{-ar_1}e^{-br_2}e^{-\lambda r_{12}}$ and that involves a 3-order derivative with respect to parameters a, b and λ. The latter were introduced in ψ_i to enable the writing of terms $r_1^{l-m}r_2^{m-n}r_{12}^{n}e^{-ar_1}e^{-br_2}e^{-\lambda r_{12}}$ appearing when using Brauner's method as derivatives (5) of D when $l-m\geq 0$, $m-n\geq 0$, and $n\geq 0$:

$$\frac{\partial^{l-m}}{\partial a^{l-m}}\left(\frac{\partial^{m-n}}{\partial b^{m-n}}\left(\frac{\partial^n D}{\partial\lambda^n}\right)\right). \quad (5)$$

For instance, the 18-parameter Hylleraas-type of [Kinoshita(1957)] requires a 9-order derivative.

Theoretically, differentiating the program that computes D is no more difficult than differentiating the related mathematical function. Each assignment statement of the code may be seen as a composition of small functions (containing one operator or one elementary function) and differentiated using the chain rule. In many areas of Science, first differentiated codes were hand coded. The work is not difficult since one uses well-known rules as: "the derivative of a sum is the sum of the derivatives". Even based on trivial rules, the manual differentiation becomes a fastidious and error prone task implying important costs of development when applied to large codes. This is especially true when implementing a high order differentiation where the number of lines of code increases exponentially with respect to the order of differentiation.
Nowadays, powerful automatic differentiation (AD) softwares are available for the differentiation of computer code, the reader is invited to visit Internet site http://www.autodiff.com/ for a review. Given an user code, the AD tool will be choose with respect to three elements.

1. **The scientific goals of the user.** Derivative computations give access to several kinds of application such as sensitivity analyses, optimisation, Hessian computations or, as discussed here, high order derivatives.

2. **The mode of differentiation.** The basic "tangent linear mode" is generally considered for sensitivity analyses giving an exact alternative to usual finite difference schemes. The so-called "adjoint mode" proposed by ODYSSÉE [Faure & Papegay(1998)], TAMC [Giering(1999)], and ADOL-C [Griewank et al.(1996)] is devoted to gradient computation involved in large-scale optimisation problems. The "high order derivative mode" constitutes an extension of the tangent linear mode. It was used in [Masmoudi & Guillaume(1996)] for the design of an optimisation process of a few parameters. Higher derivative tensors, for the evaluation of all pure and mixed partial derivatives, are discussed in [Griewank et al.(2000)].

3. **The language of programming of the user code.** The language often imposes the kind of "differentiated" code. One the one hand, the differentiation of FORTRAN codes may be tackled by "source-to-source" tools like ADIFOR [Bischof et al.(1994)], ODYSSÉE or TAMC: they analyze the source of the user code and generate the source of the differentiated code. On the other hand, the differentiation of C, C++ or FORTRAN 90 codes is generally performed at compile time by means of operator overloading (OO) libraries like ADOL-C. These libraries sometimes include the high order derivative mode of differentiation.

Existing tools propose different combinations and the user will generally encounter the one adapted to its application. Nevertheless, it remains applications, involving complex arithmetics or cross derivatives, for which no tool exists. Thus the second purpose of the paper is to provide information on the construction of an OO library allowing for the computation of high order cross derivatives of codes involving complex variables.

The outline of the paper is as follows. Section 2 presents the basis of AD as well as mathematical arguments devoted to high order cross derivative computations. OO techniques are discussed in Section 3. Sections 4 and 5 propose a set of numerical results including differentiation ones and physical ones.

2 High order cross derivative computation

Differential Calculus is one of the great achievement of Mathematics over the last centuries. It now occupies a large place in the education of many young students in Science. Some of the arguments presented here may be found in classical learning books, see for instance [Lelong-Ferrand & Arnaudies(1997)].

2.1 Basics of automatic differentiation

Given a mathematical function f, the corresponding computer code is often constructed through a decomposition of f into different parts. These are user-defined parts, specific to the problem of interest, or predefined routines devoted to integration processes, Fast Fourier Transforms or solutions of linear systems for example. From the AD point of view, the decomposition is performed up to assignment statements f_i containing one operator or one elementary function only, and control statements c_j (do-loops or conditional statements, ...). The first ones have to be differentiated, meanwhile the second ones are kept unchanged.
Differentiation of both mathematical functions and numerical codes is based on the chain rule. Nevertheless, manual differentiation often computes partial derivatives whereas AD computes Jacobians and evaluates them in directions of perturbation. This difference may be illustrated using function $f(x,y,z,w) = (x, y, xy, (xy)^2)$ which is clearly differentiable in its two variables x and y. Its main two partial derivatives evaluated at point (x_0, y_0, z_0, w_0) are:

$$\left\{ \begin{array}{l} \dfrac{\partial f}{\partial x}(x_0,y_0,z_0,w_0) = (1, 0, y_0, 2(x_0y_0)y_0), \\ \dfrac{\partial f}{\partial y}(x_0,y_0,z_0,w_0) = (0, 1, x_0, 2(x_0y_0)x_0). \end{array} \right. \tag{6}$$

In the terminology of AD, input variables x and y are said "independent" whereas output variables z and w are "dependent" ones. A variable is said to be "active" when it has a role in the differentiation. In the present case, all the variables are active ones: x and y because we differentiate f with respect to them, z and w because they depend on the first two.
Automating the differentiation, function f is first written as the compound function hog where $g(x,y,z,w) = (x,y,xy,w)$ and $h(x,y,z,w) = (x,y,z,z^2)$. The differentiation is performed evaluating the Jacobian ∇_f of f as the product of Jacobians ∇_g and ∇_h of g and h in the direction $\Delta = (\delta_x, \delta_y, \delta_z, \delta_w)^T$:

$$\begin{array}{rl} \nabla_f(x_0,y_0,z_0,w_0).\Delta & = \nabla_{hog}(x_0,y_0,z_0,w_0).\Delta, \\ & = \nabla_h(g(x_0,y_0,z_0,w_0)).\nabla_g(x_0,y_0,z_0,w_0).\Delta. \end{array} \tag{7}$$

Square matrices are required to perform such a computation, that is the reason why functions f, g and h were artificially written using the four variables. Choosing directions $(1,0,0,0)$ and $(0,1,0,0)$ one obtains equations (6). This may appear trivial, nevertheless the spirit is completely different in terms of coding. Table 1 presents the resulting derived statements written in terms of perturbations `dx`, `dy`, `dz` and `dw` that stand for perturbations δ_x, δ_y, δ_z and δ_w. In practice, the derived code also contains original statements, such as `z = x*y`, to

Table 1: Differentiated statements: `f_x` and `f_y` related to manual coding, `f_xy` related to AD.

`f_x`	`f_y`	`f_xy`
`dz = y * dx`	`dz = x * dy`	`dz = y * dx + x * dy`
`dw = 2 * z * dz`	`dw = 2 * z * dz`	`dw = 2 * z * dz`

Table 2: Recurrence relationships for the differentiation of the main operators and elementary functions. Here, terms $x_{(k)}$ and $\widetilde{x}_{(k)}$ respectively stand for the k-order derivative $x^{(k)}/k!$ and $kx^{(k)}/k!$.

operation	recurrence ($n \geq 1$)	operation	recurrence ($n \geq 1$)
$z = x + \alpha y$	$z_{(n)} = x_{(n)} + \alpha y_{(n)}$	$z = x/y$	$z_{(n)} = \dfrac{1}{y_{(0)}}\left[x_{(n)} - \displaystyle\sum_{k=0}^{n-1} z_{(k)} y_{(n-k)}\right]$
$z = xy$	$z_{(n)} = \displaystyle\sum_{k=0}^{n} x_{(k)} y_{(n-k)}$	$z = \sqrt{x}$	$z_{(n)} = \dfrac{1}{2z_{(0)}}\left[x_{(n)} - \displaystyle\sum_{k=1}^{n-1} z_{(n)} z_{(n-k)}\right]$
$z = e^x$	$\widetilde{z}_{(n)} = \displaystyle\sum_{k=1}^{n} z_{(n-k)} \widetilde{x}_{(k)}$	$z = \ln(x)$	$\widetilde{z}_{(n)} = \dfrac{1}{x_{(0)}}\left[\widetilde{x}_{(n)} - \displaystyle\sum_{k=1}^{n-1} x_{(n-k)} \widetilde{z}_{(k)}\right]$

enable a correct evaluation.
Generalizing the purpose to a large code `C`, the manual method requires the writing of two derived codes `C_x` and `C_y` evaluated implicitly using canonical directions, whereas the second one leads to the writing of a unique code `C_x_y` evaluated in directions specified by the user. Although numerical results are identical, the second method was retained by AD developers and most of scientists that hand coded the differentiation of their codes. On the contrary, the first approach applied to high order cross derivative computations, will multiply the parts of derived code.

2.2 High order differentiation

Optimised recurrence relationships were deduced from the Differential Calculus theory. For example the Leibniz formula (8) enables the n-order differentiation of the multiplication:

$$(xy)_{(n)} = \sum_{k=0}^{n} x_{(k)} y_{(n-k)}, \tag{8}$$

where $x_{(k)}$ denotes here the k-order derivative $x^{(k)}/k!$ of x. Written in such a manner they are of easy implementation, especially when developing an AD tool based on the OO technique. These relationships do not depend on the kind of arithmetics, they still applies on complex variables.

Unfortunately, there does not exist a simplified optimised recurrence formula for the hypergeometric Gauss series. Brauner's method makes use the hypergeometric function ${}_2F_1$ instead of the ${}_1F_1$ function appearing in (4). This ${}_2F_1$ function is evaluated on the convergence circle as:

$$ {}_2F_1(\alpha_1,\alpha_2;\alpha_3;z)=\sum_{n=0}^{\infty}\frac{(\alpha_1)_n(\alpha_2)_n}{(\alpha_3)_n}.\frac{z^n}{n!},\quad z\in\mathbb{C},\ |z|<1, \tag{9}$$

where the parameters α_i $(i=1,..,3)$ are computed by a recurrence formula as:

$$ (\alpha_i)_0=1,\ \text{and}\ (\alpha_i)_n=\alpha_i(\alpha_i+1)..(\alpha_i+n-1)=\frac{\Gamma(\alpha_i+n)}{\Gamma(\alpha_i)},\ \forall n\geq 1. \tag{10}$$

A fair implementation of ${}_2F_1$ has to take the convergence properties into account. When series (9) is a convergent one, a recurrence formula allows for its differentiation with respect to z:

$$ \frac{d^n}{dz^n}({}_2F_1(\alpha_1,\alpha_2;\alpha_3;z))=\frac{(\alpha_1)_n(\alpha_2)_n}{(\alpha_3)_n}.{}_2F_1(\alpha_1+n,\alpha_2+n;\alpha_3+n;z). \tag{11}$$

Consequently, function ${}_2F_1$ does not need to be differentiated explicitly: recursive calls are sufficient. As explained before, formula (11) is incomplete from an AD point of view because variable z represents much more than a mathematical variable. It has to be seen as a function $z(x)$, depending on some active variable x of the code, to enable a correct use of the chain rule. Thus the first order differentiation formula writes as follows:

$$ \frac{d}{dx}({}_2F_1(\alpha_1,\alpha_2;\alpha_3;z(x)))=\frac{\alpha_1\alpha_2}{\alpha_3}.{}_2F_1(\alpha_1+1,\alpha_2+1;\alpha_3+1;z(x)).\frac{dz(x)}{dx}. \tag{12}$$

Since (12) involves ${}_2F_1$ functions evaluated with different parameters, no cheap recurrence formula exists. High order derivatives of compound hypergeometric function ${}_2F_1(a,b;c;z(x))$ are calculated from the formula proposed by [Faá di Bruno (1857)]:

$$ \begin{aligned}\frac{\partial^n {}_2F_1(a,b;c;z(x))}{\partial x^n}=\sum_{\substack{\gamma_1,..,\gamma_n\\ s.t.\ \beta=n}} & \frac{n!}{\gamma_1!..\gamma_n!}\frac{(a+\gamma)_n(b+\gamma)_n}{(c+\gamma)_n}.(z_{(1)}^{\gamma_1}(x)..z_{(n)}^{\gamma_n}(x)).\\ & {}_2F_1(a+\gamma,b+\gamma;c+\gamma;z(x)),\end{aligned} \tag{13}$$

where $\gamma=\sum_{j=1,..,n}\gamma_j$, and $\beta=\sum_{j=1,..,n}j\gamma_j$. The latter represents the sum over all the integer partitions of n and grows exponentially with n: there exist 11 partitions for $n=6$ and 30 partitions for $n=9$.

2.3 High order cross derivative computation

The n-order derivative ∇_f^n of a n-differentiable function $f(x,y)$ with respect to x and y evaluated in direction (δ_x, δ_y) may be written as:

$$\frac{1}{n!}\nabla_f^n(x,y).(\delta_x,\delta_y)^n = \sum_{k=0}^{n} \frac{\delta_x^k}{k!}.\frac{\delta_y^{n-k}}{(n-k)!}.\frac{\partial^n f}{\partial x^k \partial y^{n-k}}(x,y). \tag{14}$$

AD philosophy designs the high order differentiation as a process devoted to the computation of ∇_f^n without considering formula (14). When implementing relationships presented in Table 2, the computation of $\nabla_f^n.(\delta_x,\delta_y)^n$ is done recursively: right hand side terms $\frac{\partial^n f}{\partial x^k \partial y^{n-k}}$ are not calculated individually as in a manual differentiation. However, the knowledge of such cross derivatives are required when using Brauner's method for the study of the ionisation. These are not computable in a straightforward manner from AD point of view. But one may exploit the linearity property of the differentiation in order to extract the cross derivatives of interest. For instance derivative $\frac{\partial^2 f}{\partial x \partial y}$ may be deduced using formula (14) (with $n=2$) as $\left(\nabla_f^2.(1,1)^2 - \nabla_f^2.(1,0)^2 - \nabla_f^2.(0,1)^2\right)/2$ or as $\left(\nabla_f^2.(1,1)^2 - \nabla_f^2.(1,-1)^2\right)/2$. As proved below, any n-order cross derivative of f depending on two variables may be deduced from a linear combination of $n+1$ terms ∇_f^n evaluated in well-chosen directions of perturbation.

Proposition: *Any n-order cross derivative $\frac{\partial^n f}{\partial x^k \partial y^{n-k}}$, $0 \le k \le n$, may be computed using at most $n+1$ non-collinear directions of perturbation.*

Proof: Although the proof is trivial (one uses the linearity property of the differentiation), we present it to point out the need of an exact solution for the underlying linear system.

For the sake of simplicity in the writing of theoretical developments, we decided to replace (14) by the polynomial formula:

$$\frac{1}{n!}(x+y)^n.(\delta_x,\delta_y)^n = \sum_{k=0}^{n} \frac{\delta_x^k}{k!}.\frac{\delta_y^{n-k}}{(n-k)!}.x^k y^{n-k}, \tag{15}$$

identifying formally $\frac{\partial^n f}{\partial x^k \partial y^{n-k}}(x,y)$ to $x^k y^{n-k}$. One obtains the binomial formula when choosing the direction of perturbation $(\delta_x,\delta_y)=(1,1)$.

Let $\{\Delta_p\}_{p=0,..,n} = \{(1,\delta_p)\}_{p=0,..,n}$ be a set of $n+1$ non-collinear directions of perturbation. Writing formula (15) for each direction δ_p one obtains a linear system of $n+1$

equations depending on $n+1$ unknowns. Written under a matrix form, one has:

$$\begin{pmatrix} (x+y)_n.(1,\delta_0)^n \\ \vdots \\ (x+y)_n.(1,\delta_{n-p})^n \\ \vdots \\ (x+y)_n.(1,\delta_n)^n \end{pmatrix} = \begin{pmatrix} 1 & .. & \delta_0^{n-p} & .. & \delta_0^n \\ \vdots & :: & \vdots & :: & \vdots \\ 1 & .. & \delta_{n-p}^{n-p} & :: & \delta_p^n \\ \vdots & :: & \vdots & :: & \vdots \\ 1 & .. & \delta_n^{n-p} & :: & \delta_n^n \end{pmatrix} . \begin{pmatrix} x_0 y_n \\ \vdots \\ x_p y_{n-p} \\ \vdots \\ x_n y_{n-n} \end{pmatrix}, \tag{16}$$

where the condensed notation x_k stands for $x^k/k!$ to get rid of factorial coefficients. One recognizes a Van der Monde matrix $\mathbf{V}$ which determinant is positive since Δ_i and Δ_j $(j \neq i)$ are non-collinear.
In practice, monomials x^n and y^n, that stand for partial derivatives $\dfrac{\partial^n f}{\partial x^n}$ and $\dfrac{\partial^n f}{\partial y^n}$, are obtained using directions of perturbation $(1,0)$ and $(0,1)$. Terms $x^k y^{n-k}$ $(k = 1,..,n-1)$ cannot be computed in a straightforward manner. These may be deduced as linear combinations of polynomials $(x+y)^n.(1,\delta_p)^n$ $(p = 0,..,n)$ that represent AD-computable derivatives $\nabla^n(f).(1,\delta_p)^n$. One solves problems such as:

$$Find\ \Upsilon_k = (\varepsilon_0,..,\varepsilon_n)\ s.t.\ \sum_{p=0}^{n} \varepsilon_p (x+y)^n.(1,\delta_p)^n = x^k y^{n-k}, \quad \forall x,y. \tag{17}$$

Choosing $x = y = 1$, one deduces that Υ_k is solution of a linear system involving Van der Monde matrix $\mathbf{V}$, and which right hand side term is chosen as the $k+1^{th}$ vector of the canonical basis of $\mathbb{R}^{n+1}$.
end of the proof

Problem (17) has to be solved exactly to get exact derivatives because the use of an iterative scheme with a precision of 10^{-6} for example will limit derivative calculations to the same precision. This is not enough when studying electron-atom collision. In practice, a convenient set of directions of perturbation is $S = \{(1,0),(1,1),(1,-1),(1,2),(1,-2),...\}$ since it eases manual solutions of (17).
Table 3 presents the set of directions of perturbation we use for the computation of cross derivatives appearing the Kinoshita wavefunction ψ_K involving 18 parameters:

$$\begin{aligned} \psi_K(s,u,t) = [\quad & c_{0,0,0} + c_{1,0,0}s + c_{1,1,0}u + c_{2,0,0}s^2 + c_{2,1,0}su + c_{2,2,0}u^2 + \\ & + c_{2,2,2}t^2 + c_{3,0,0}s^3 + c_{3,3,0}u^3 + c_{3,2,2}st^2 + c_{3,3,2}ut^2 + \\ & + c_{4,0,0}s^4 + c_{4,4,0}u^4 + c_{4,2,2}s^2t^2 + c_{4,4,2}u^2t^2 + c_{5,5,0}u^5 + \\ & + c_{5,5,2}u^3t^2 + c_{6,6,2}u^4t^2]e^{-ks/2}. \end{aligned} \tag{18}$$

It was chosen because it has no negative-power terms. Wave function ψ_K depends on variables s, t, and u, but each monomial depends on a couple of variables only. The theoretical results may be applied. The directions of perturbations are triplets: $\delta_s = (1,1,0)$ is equal to the sum of $\delta_a = (1,0,0)$ and $\delta_b = (0,1,0)$, $\delta_t = (1,-1,0) = \delta_a - \delta_b$, and $\delta_u = \delta_\lambda = (0,0,1)$.

one will choose the more convenient set of variables – $\{a, b, \lambda\}$ or $\{s, t, u\}$ – for the computation with a given wavefunction.

From Table 3 it can be seen that the partial derivative $\dfrac{\partial^6 D}{\partial t^2 \partial u^4}$ may be computed as:

$$\begin{aligned}\frac{\partial^6 D}{\partial t^2 \partial u^4} = \frac{1}{360}\Big[\quad & 96\nabla_D^6.\delta_t^6 - 30\nabla_D^6.\delta_u^6 + 16\nabla_D^6.\delta_{t+u}^6 + \\ & +16\nabla_D^6.\delta_{t-u}^6 - \nabla_D^6.\delta_{2t+u}^6 - \nabla_D^6.\delta_{2t-u}^6\Big].\end{aligned} \tag{19}$$

The Kinoshita wavefunction is computable using 10 directions only. One observes that any k-order partial derivatives ($k = 1$ and $k \geq 3$) are calculable using k directions of perturbation, that is k evaluations of the overloaded code. This remains true for $k = 2$ when choosing directions δ_{s+u} and δ_{s-u}, but this choice implies one direction more (δ_{s-u}) and consequently a loss of 10% in the efficiency of the code.

Cross derivative involving m variables may also be tackled. One uses the general multinomial formula:

$$(\delta_{x1}x_1 + .. + \delta_{xm}x_m)^n = \sum_{n_1+..+n_m=n} \frac{n!}{n_1!..n_m!}(\delta_{x1}^{n_1}..\delta_{xm}^{n_m}){x_1}^{n_1}..{x_m}^{n_m}, \tag{20}$$

that stands for the generalized n-order derivative of $f(x_1, .., x_m)$ evaluated with direction $(\delta_{x1}, .., \delta_{xm})$:

$$\begin{aligned}\nabla_f^n(x_1, .., x_m).(\delta_{x1}, .., \delta_{xm})^n = \sum_{n_1+..+n_m=n} \quad & \frac{n!}{n_1!..n_m!}(\delta_{x1}^{n_1}..\delta_{xm}^{n_m}). \\ & .\frac{\partial^n f}{\partial x_1^{n_1}..\partial x_m^{n_m}}(x_1, .., x_m).\end{aligned} \tag{21}$$

The number of non-collinear directions of perturbation required to identify each partial derivative is equal to the number $\mathcal{N}(m, n)$ of combinations $(n_1, .., n_m)$ of integers such that $n_1 + .. + n_m = n$ $(0 \leq n_i \leq n)$. A small program suffices to evaluate $\mathcal{N}$. One observes that the complexity of $\mathcal{N}(m, n)$ is of $O(n^{m-1})$. Sequences are given in [Sloane(2003)].

In the particular case of three variables, one has $\mathcal{N}(3, n) = (n + 1)(n + 2)/2$ for $n \geq 3$ (triangular number sequence), that is $\mathcal{N}(3, 3) = 10, \mathcal{N}(3, 6) = 28, \mathcal{N}(3, 9) = 55, ...$ Choosing following non-collinear directions, one may compute $\dfrac{\partial^3 f}{\partial a \partial b \partial \lambda}$ from derivatives ∇^3 as:

$$\begin{aligned}6\frac{\partial^3 f}{\partial a \partial b \partial \lambda} = \quad & \nabla_f^3.(1,1,1)^3 - \nabla_f^3.(1,1,0)^3 - \nabla_f^3.(1,0,1)^3 - \nabla_f^3.(0,1,1)^3 + \\ & +\nabla_f^3.(1,0,0)^3 + \nabla_f^3.(0,1,0)^3 + \nabla_f^3.(0,0,1)^3.\end{aligned} \tag{22}$$

Table 3: Directions of perturbation for Kinoshita wavefunction.

monom.	coef	δ_s $\begin{pmatrix}1\\1\\0\end{pmatrix}$	δ_t $\begin{pmatrix}-1\\1\\0\end{pmatrix}$	δ_u $\begin{pmatrix}0\\0\\1\end{pmatrix}$	δ_{s+u} $\begin{pmatrix}1\\1\\1\end{pmatrix}$	δ_{t+s} $\begin{pmatrix}0\\2\\0\end{pmatrix}$	δ_{t-s} $\begin{pmatrix}-2\\0\\0\end{pmatrix}$	δ_{t+u} $\begin{pmatrix}-1\\1\\1\end{pmatrix}$	δ_{t-u} $\begin{pmatrix}-1\\1\\-1\end{pmatrix}$	δ_{2t+u} $\begin{pmatrix}-2\\2\\1\end{pmatrix}$	δ_{2t-u} $\begin{pmatrix}-2\\2\\-1\end{pmatrix}$
1	1	1	0	0	0	0	0	0	0	0	0
s	1	1	0	0	0	0	0	0	0	0	0
s^2	1	1	0	0	0	0	0	0	0	0	0
s^3	1	1	0	0	0	0	0	0	0	0	0
s^4	1	1	0	0	0	0	0	0	0	0	0
t^2	1	0	1	0	0	0	0	0	0	0	0
u	1	0	0	1	0	0	0	0	0	0	0
u^2	1	0	0	1	0	0	0	0	0	0	0
u^3	1	0	0	1	0	0	0	0	0	0	0
u^4	1	0	0	1	0	0	0	0	0	0	0
u^5	1	0	0	1	0	0	0	0	0	0	0
su	1/2	-1	0	-1	1	0	0	0	0	0	0
st^2	1/6	-2	0	0	0	1	-1	0	0	0	0
s^2t^2	1/12	-2	-2	0	0	1	1	0	0	0	0
t^2u	1/6	0	0	-2	0	0	0	1	-1	0	0
t^2u^2	1/12	0	-2	-2	0	0	0	1	1	0	0
t^2u^3	1/240	0	0	-30	0	0	0	16	-16	-1	1
t^2u^4	1/360	0	96	-30	0	0	0	16	16	-1	-1

3 Operator overloading library

AD proposes two kinds of software for the differentiation of a code. On the one hand, the source-to-source transformation may be seen as a mimic of manual differentiation. A second order code is still writable [Charpentier et al.(2002)] but AD source-to-source softwares were not designed for this. The higher the order of differentiation, the larger the number of derived statements. This also explains why hand-coding was limited to the 5^{th} order in [Ancarani et al.(2004)]. On the other hand, OO is part of the abilities of oriented-objected languages such C++ or FORTRAN 90. Basically it consists in the definition of new data structures. Operations are associated to these structures redefining the usual operators of the programming language. OO keeps unchanged the assignment and control statements of the code, the differentiation being implemented through the recurrence formulas of Table 2 in an external library. A C-implementation of an OO library for the tangent linear mode of differentiation is presented with details in [Griewank(2000)].
The AD tool we need has to satisfy several constraints.

1. **Complex arithmetics.** None of the existing AD tools deals with this kind of code even usual rules still apply.
2. **High order derivatives.** This strongly suggests the use of OO techniques that are well adapted to the implementation of recurrence relationships, hiding details of programmation in a library constructed once for all. Basics of OO are presented below.
3. **Differentiation of the hypergeometric function.** A recurrence formula (13) exists. This justifies OO again.
4. **Crossed derivatives.** None of existing AD tools propose this kind of differentiation.
5. FORTRAN 90 **library.** The user code is written in FORTRAN 77.

When working with a FORTRAN code written in complex arithmetics, variables that may have a role in the differentiation are of type `REAL*8` and `COMPLEX*16` (or eventually `REAL*4` and `COMPLEX*8`). Thus, our library defines two new types in a `module`, namely `dr8` and `dc16`, that include differentiation fields up to the required order n. Given a variable `a` of type `dr8`, the library (for $n = 2$) considers that `a%d0`, `a%d1` and `a%d2` respectively host the `REAL*8` value of variable `a` computed as in the original code, and the `REAL*8` values of first and second order derivatives. By means of a set of `modules`, the library overloads the classical operators (`+`, `-`, `*`, `/` and `**`) and the elementary functions (trigonometric, exponential and logarithmic ones) for both types `dr8` and `dc16`. Codes may also contain user-defined mathematical functions that need to be "overloaded" as well.
The differentiation of any FORTRAN code through the OO library is performed introducing the `modules` names at the top of the routines and defining the independent variables changing their type to `dr8` for real variables and `dc16` for complex variables. The code being differentiated with respect to them, the user has to provide their first order fields `%d1` with the directions of perturbation he chooses. According to the chain rule, the active dependent

variables need to change their type. One detects them at compile time since the assignment symbol = is not overloaded: the left hand side term of any assignment statement has to be of compatible type with the types of variables appearing in the right hand side. What could be seen as a lack of the library allows for the follow-up of the active variables into the code. Once the propagation of the differentiation is correct, the code enables the computation of high order derivatives, high order cross derivatives being obtained from linear combinations computed once for all.

Table 4: Taylor test on D in the direction δ_a. Only the first 12 digits are printed.

$10^{-\omega}$	ν_ω^{Re}	ν_ω^{Im}	$10^{-\omega}$	ν_ω^{Re}	ν_ω^{Im}
0	0.7222726554	0.5201929826	8	1.0000000361	1.0000000067
1	0.9637322971	0.9309441517	9	1.0000001374	1.0000046761
2	0.9962602546	0.9928022931	10	1.0000072285	1.0000406879
3	0.9996248591	0.9992772003	11	1.0000781393	1.0000880718
4	0.9999624746	0.9999276900	12	1.0001794404	1.0028736725
5	0.9999962477	0.9999927693	13	1.0028808028	1.0223441596
6	0.9999996262	0.9999992764	14	1.0467779423	1.1975211083
7	0.9999999618	0.9999999665	15	1.0130109119	2.9866233877

4 Numerical results

The differentiation proposed by the library may be verified in two steps: a "Taylor test" and a recursive validation of successive derivatives.
The usual "Taylor test" consists in a first order comparison of the derivatives obtained by differentiation with a finite difference computation. In the case of a complex function $f(x)$, the following two ratios may be computed:

$$\nu_\omega^{Re} = \frac{|Re(f(x+\omega\delta_x) - f(x))|}{\omega|Re(\nabla_f(x).\delta_x)|}, \text{ and } \nu_\omega^{Im} = \frac{|Im(f(x+\omega\delta_x) - f(x))|}{\omega|Im(\nabla_f(x).\delta_x)|}, \tag{23}$$

where x is the active variable (real or complex) of differentiation, δ_x is a real direction of perturbation, ω is the step of discretisation of the finite difference method, and ∇_f is the differentiated function. The differentiation is correct when ratios (23) tend linearly towards 1 as ω tends linearly to 0. As observed numerically in Table 4, small steps ω are necessary to minimize errors coming from the truncation of the Taylor expansion, $\omega = 10^{-8}$ being the optimal value. Then, the subtraction of too close floating-point numbers leads to a cancellation error large enough to dominate the truncation error. It is well-known that such verifications loose their accuracy for higher order derivatives.
Fortunately, high order derivatives may be verified recursively. The library is applied to the computations D and its manual derivative D_a. As a consequence, one compares derivatives $\frac{\partial^n D}{\partial a^n}$ with derivatives $\frac{\partial^{n-1} D_a}{\partial a^{n-1}}$ for increasing $n \geq 1$. Real parts of these two terms are shown in Table 5, they are similar up to 14 digits, that corresponds to the numerical precision of the computer. Imaginary parts are also computed and similar results could be reported.

Performances are evaluated according to the computational methods involved in this study.

1. **Number of operations.** Whatever is the direction of perturbation, a n-order differentiation performed through recurrence relationships (Table 2) have a complexity

Table 5: Real parts of successive derivatives of $D = \dfrac{\partial^0 D}{\partial a^0}$ and $D_a = \dfrac{\partial^0 D_a}{\partial a^0}$.

n	$Re\left(\dfrac{\partial^n D}{\partial a^n}\right)$	$Re\left(\dfrac{\partial^{n-1} D_a}{\partial a^{n-1}}\right)$
0	1.3560574723339032E-3	
1	-6.421661364219621E-4	-6.42166136421962E-4
2	4.8197314270524656E-4	4.819731427052466E-4
3	-5.013083275497681E-4	-5.013083275497683E-4
4	6.622768551827872E-4	6.622768551827877E-4
5	-1.047620568979999E-3	-1.0476205689799995E-3
6	1.9070566076086795E-3	1.907056607608683E-3

in n^2 [Griewank(2000)]. But, the collision code includes hypergeometric functions for which no cheap recurrence formula exists. Although the exponential growth of the number of partitions is slow, the complexity is stronger than a quadratic one.

When dealing with wavefunction (18), a 6-order differentiation of D involves 11 partitions, where as a 9-order differentiation of the start `C` code would involve 30 partitions. Managing computations from D reduces in a important manner the cost of computation of the derivatives of ${}_2F_1$.

2. **Number of directions of perturbation.** The complexity is linear (resp. quadratic) when a code is differentiated with respect to two (resp. three) variables.

   Working with D, wavefunction (18) requires 10 directions (Table 3) because the differentiation is performed with respect to 3 couples of independent variables. One notices that cross derivatives of the start code `C` involves 3 independent variables. Such calculation has a complexity in n^2: extracting a 9-order derivative requires more or less $\mathcal{N}(3,9) = 55$ directions.

3. **Number of integrals.** Brauner's method involves a double integral, the ${}_2F_1$ function and a certain number of derivative computation whereas the method used in [Jones & Madison(2003)] needs a 6-dimensional quadrature with 3 calls to the ${}_1F_1$ function. This 6-dimensional integral was studied in [Rasch et al.(1998)]. The CPU time was around 3360 seconds on 64 nodes of the Hitachi supercomputer SR2201 at Cambridge (256 nodes, 150MHz/node) while Brauner's method (with a 6-order derivative of D and a 2-dimensional integral) needs less than 3 hours on one node of the IBM POWER 3 of CINES (16 nodes, 375MHz/node).

Obviously, the computations realized from D (10*11*4 ${}_2F_1$ computations because D already involves 4 ${}_2F_1$) is much cheaper to achieve than a computation performed from the start code `C` (30*55 ${}_2F_1$ computations).

Without any doubt, OO is a precious technique for the computation of high order derivatives. It remains true when cross derivatives with respect to two independent variables are considered. The library may be applied to any differentiable wavefunction, the performance depending on the order of differentiation and the number of independent variables.

5 Ionisation results

The energy of the ground state of helium is E=-2.903724 a.u.. Figures 1 and 2 compare the five-fold differential cross section (FDCS) calculated for three different initial wavefunctions:

1. the Kinoshita wavefunction ($E_K = -2.903715 a.u.$) written in (18),
2. the [Stewart & Webb(1963)] wavefunction ($E_{SW} = -2.90332 a.u.$):

$$\psi_{SW}(s,u,t) = e^{-ks/2}(c_0 + c_1 u + c_2 t^2 + c_3 s + c_4 s^2 + c_5 u^2), \tag{24}$$

3. the [Bonham & Kohl(1966)] wavefunction ($E_{BK} = -2.903115 a.u.$):

$$\begin{aligned}\psi_{BK}(r_1,r_2,r_{12}) = \quad & e^{-a_1 r_1 - a_2 r_2} + e^{-a_2 r_1 - a_1 r_2} \\ & + (e^{-a_3 r_1 - a_4 r_2} + e^{-a_4 r_1 - a_3 r_2}) b_2 e^{-r_{12}}.\end{aligned} \tag{25}$$

On the one hand, curves are quite identical whatever is the accuracy of the initial wavefunction. Increasing the number of derivatives has no effect on the FDCS: Brauner's method is a convergent one. On the other hand, the agreement with the data of [Lahmam-Bennani et al.(1999)] is not complete since computations reproduce the shape only: there still exists a factor 2 between experiments and theory. Curiously, wavefunction $\psi_i = e^{-a(r_1+r_2)}(1 + .25 r_{12} e^{r_{12}/4})$ with a bad energy value $E = 2.8766 a.u.$ was able to reproduce the magnitude of the experimental results [Ancarani et al.(2004)].

Many questions are still open for the interpretation of the $(e, 3e)$ measurements on helium at 5.6 keV incident energy because the models proposed in the literature are unable to reproduce the magnitude of these experiments. Indeed, none of the models propose an accurate treatment of the TS1 mechanism that may occur in a double ionisation. In this mechanism, the interaction between the incoming electron and one of the target electrons leads to a first ejected electron. Thus, its interaction with the other target electron sometimes induces its ejection. The second Born approximation is able to describe the two interactions of the TS1 mechanism since two interactions are involved. Nevertheless, this second Born approximation requires a 6-dimensional quadrature when applying Brauner's method, and a 10-dimensional quadrature if the method of Jones and Madison is used. Obviously, our method could be used while the method of Jones and Madison, that requires a larger amount of CPU time, would reach the limits of most of supercomputers.

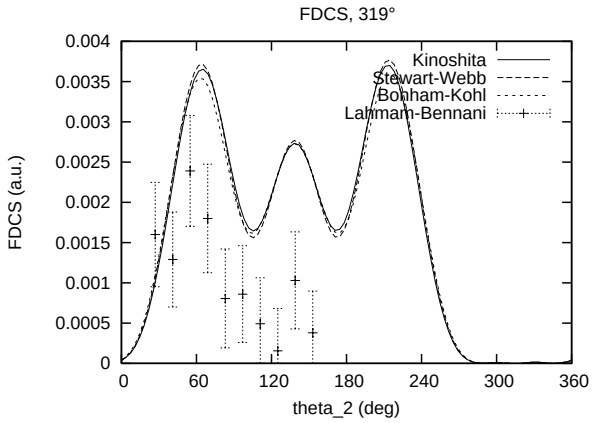

Figure 1: FDCS for $(e, 3e)$ ionisation of the helium ground state, as a function of the angle θ_2. The other ejected electron is detected at $\theta_1 = 319^{\mathrm{O}}$. Vertical bars: absolute experimental data [Lahmam-Bennani et al.(1999)]. Solid line: Kinoshita wavefunction. Dashed line: Stewart-Webb wavefunction. Dotted line: Bonham-Kohl wavefunction.

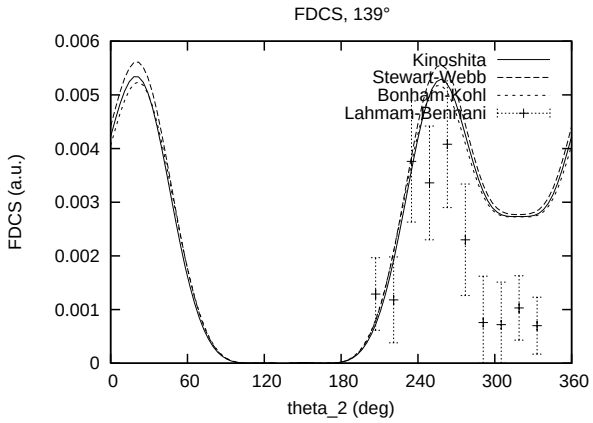

Figure 2: FDCS for $(e, 3e)$ ionisation of the helium ground state, as a function of the angle θ_2. The other ejected electron is detected at $\theta_1 = 139^{\mathrm{O}}$. Vertical bars: absolute experimental data [Lahmam-Bennani et al.(1999)]. Solid line: Kinoshita wavefunction. Dashed line: Stewart-Webb wavefunction. Dotted line: Bonham-Kohl wavefunction.

6 Conclusions

The OO offers a precious solution to achieve high order differentiation, cross derivatives being deduced. This especially true for differentiation with respect to two independent variables as proved by the FDCS calculation (figure 2) involving up to a 6-order differentiation of the generic term D. Complexity calculations give information on the actual feasibility of cross derivative computation assessing how the order of differentiation and the number of independent variable could limit the performance.

The library we constructed in FORTRAN 90 for our collision application, includes the differentiation of the hypergeometric function ${}_2F_1$. Two more classical version (dealing with operators and elementary functions only) are under development in FORTRAN 90 and C++. They allow for some automatic optimized computation of cross derivatives involving two independent variables through the management of linear combinations of directions of S. They will be documented in a next future.

The model of first Born approximation presented in this paper describes one of the mechanisms of the double ionisation, namely the shake-off mechanism. The latter consists of a collision between the incoming electron and the target. This first ionisation is followed by a relaxation process due to the sudden change of potential that is responsible for a second ejection. Calculations are realized with an accurate description of the initial state of the target. Based on a 2-dimensional integral and computations of high order cross derivatives, our method gives the same results than the method [Jones & Madison(2003)] that deals with a very expensive 6-dimensional integral. Furthermore, numerical results are compared with the recent experiments of [Lahmam-Bennani et al.(1999)]. It appears that the accurate Kinoshita wavefunction does not improve the results obtained with the lighter Bonham-Kohl wavefunction: the model still reproduces the shape but not the magnitude. The second mechanism (TS1) could be tackled in a next future with our method using the same description of the initial state and of the final state.

Acknowledgments: The authors thank the Centre Informatique National de l'Enseignement Supérieur (CINES) for providing free computer time. The help from G. Zenteno at Instituto de Geología, UNAM, is also gratefully acknowledged.

References

[Ancarani et al.(2004)] Ancarani, L.U., Montagnese, T., & Dal Cappello, C., 2004, Role of the helium ground state in $(e, 3e)$ processes, Phys. Rev. A, **70**, 1–10.

[Berakdar et al.(2003)] Berakdar, J., Lahmam-Bennani,A. , & Dal Cappello, C., 2003, The electron impact double ionization of atoms : an insight into the four-body Coulomb scattering dynamics, Phys. Rep., **374**, 91–164.

[Bischof et al.(1994)] Bischof, C., Carle, A., Khademi, P., & Mauer, A., 1994, The ADIFOR2.0 System for the Automated Differentiation of Fortran 77 Programs, Argonne Preprint ANL-MCS-P481-1194 and CRPC Technical Report CRPC-TR94491.

[Bonham & Kohl(1966)] Bonham, R.A., & Kohl, D.A., 1966, Simple correlated wavefunctions for the ground state of helium like atoms, J. Chem. Phys., 45, 2471–2473.

[Brauner et al.(1989)] Brauner, M., Briggs, J., & Klar, H., 1989, Triply-differential cross sections for ionisation of hydrogen atoms by electrons and positrons, J. Phys. B: At. Mol. Phys., **22**, 2265–2287.

[Charpentier et al.(2002)] Charpentier, I., Jakse, N., & Veersé, F., 2002, Second Order Exact Derivatives to Perform Optimization on Self-Consistent Integral Equations Problems, Proceedings of Automatice Differentiation 2001: From Simulation to Optimization, Springer-Verlag, 189–197.

[Faá di Bruno (1857)] Faá di Bruno, F., 1857, Note sur une nouvelle formule de calcul differentiel, Quart. J. Pure Appl. Math., **1**, 359–360.

[Faure & Papegay(1998)] Faure, C., & Papegay, Y., 1998, Odyssée User's guide. Version 1.7, Technical Report INRIA RT-211.

[Giering(1999)] Giering, R., 1999, Tangent linear and Adjoint Model Compiler , Users manual 1.4.

[Griewank et al.(1996)] Griewank, A., Juedes, D., Srinivasan, J., & Tyner, C., ADOL-C, A Package for the Automatic Differentiation of Algorithms Written in C/C++, Algor.755, ACM Transactions on Mathematical Software, **22**, 131–167.

[Griewank(2000)] Griewank, A., 2000, *Evaluating derivatives: Principles and techniques of algorithmic differentiation*, Frontiers in Appl. Math. Number 19, SIAM (Penn., 2000).

[Griewank et al.(2000)] Griewank, A., Utke, J. and Walther, A., Evaluating higher derivative tensors by forward propagation of univariate Taylor series, Mathematics of computation, **69**, 1117–1130.

[Jones & Madison(2003)] Jones, S., & Madison, D.H., 2003, Single and double ionization of atoms by photons, electrons, and ions, AIP Conference proceedings 697, Hanne, G.F., Malegat, L., & Schmidt-Böcking, H. (Eds), 70–73.

[Joulakian et al.(1992)] Joulakian, B., Dal Cappello, C., & Brauner, M., 1992, Double ionization of helium by fast electrons: use of correlated two electron wavefunctions, J. Phys. B: At. Mol. Phys., **25**, 2863–2871.

[Kheifets et al.(2002)] Kheifets, A., Bray, I., Berakdar, J., & Dal Cappello, C., 2002, Comparative theoretical study of $(e, 3e)$ on helium: Coulomb-waves versus close-coupling approach, J. Phys. B: At. Mol. Phys., **35**, L15–L21.

[Kinoshita(1957)] Kinoshita , T., 1957, Ground state of the helium, Phys. Rev., **105**, 1490–1502.

[Lahmam-Bennani et al.(1999)] Lahmam-Bennani, A., Taouil, I., Duguet, A., Lecas, M., Avaldi, L., & Berakdar, J., 1999, Origin of dips and peaks in the absolute fully resolved cross sections for the electron-impact double ionisation of helium, Phys. Rev. A, **59**, 3548–3555.

[Lelong-Ferrand & Arnaudies(1997)] Lelong-Ferrand, J., & Arnaudies, J.M., 1997, Cours de Mathématiques, Tome 2: Analyse, Dunod, 648p.

[Masmoudi & Guillaume(1996)] Masmoudi, M. & Guillaume, P., 1996, Computation of high order derivatives in optimal shape design, Numerishe Mathematik, **67**, 831–866.

[Rasch et al.(1998)] Rasch, J., Whelan, C.T., Lucey, S.P., Dal Cappello, C., & Walters, H.R.J., 1998, 6-dimensional integrals and supercomputers, Comput. Phys. Commun., **114**, 378–384.

[Stewart & Webb(1963)] Stewart & Webb, 1963, Photo-ionisation of helium and ionised lithium, Proc. Phys. Soc., **82**, 532–536.

[Sloane(2003)] Sloane, N.J.A., 2003, The On-Line Encyclopedia of Integer Sequences, http://www.research.att.com/ njas/sequences/.

Contents

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399